WHAT IS CLAIMED IS:

1. A client-server relational database system, comprising:

a client computer;

a server computer; and

a network connecting the client computer and the server computer;

wherein data from the client computer is encrypted by the client computer and hosted by the server computer, the encrypted data is operated upon by the server computer to produce an intermediate results set, the intermediate results set is sent from the server computer to the client computer where it is operated upon by the client computer and then returned to the server computer where it is further operated upon by the server computer before being sent again from the server computer to the client computer in order to produce actual results.

2. The system of claim 1, wherein the client computer decrypts the intermediate results set, performs one or more operations on the decrypted intermediate results set to generate an updated intermediate results set, re-encrypts the updated intermediate results set, and returns the re-encrypted intermediate results set to the server computer.

3. The system of claim 2, wherein the operations comprise logical comparison operations.

4. The system of claim 2, wherein the operations comprise filtering operations.

5. The system of claim 2, wherein the operations comprise sorting operations.

6. The system of claim 1, wherein the server computer executes a round-trip filtering operator that specifies when the intermediate results set is sent from the server computer to the client computer to be operated upon by the client computer and then returned to the server computer to be operated upon by the server computer.

7. The system of claim 6, wherein the round-trip filtering operator sends maybe tuples from the server computer to the client computer, the client computer filters out certain tuples from the maybe tuples, and the server computer receives back only certain tuples from the client computer.

8. The system of claim 6, wherein the server receives a query from the client computer, generates a plurality of query execution plans having different placements of the round-trip filtering operator, and chooses one of query execution plans that optimizes placement of the round-trip filtering operator.

9. The system of claim 6, wherein the server executes a last-trip decryption operator that specifies when the intermediate results set is sent from the server computer to the client computer in order to produce actual results.

10. The system of claim 9, wherein the server computer receives a query from the client computer, generates a plurality of query execution plans having different placements of the last-trip decryption operator, and chooses one of query execution plans that optimizes placement of the last-trip decryption operator.

11. The system of claim 10, wherein the server computer optimizes placement for the round-trip filtering operators and then optimizes placement for the last-trip decryption operator.

12. The system of claim 9, wherein the last-trip decryption operator can be pulled up above any unary and binary operator in a query tree, except for a GroupBy operator.

13. The system of claim 12, wherein the GroupBy operator can be pulled above a unary operator other than another GroupBy operator if and only if all columns used in the unary operator are functionally computed by grouping columns of an input relation.

14. The system of claim 13, wherein the GroupBy operator can be pulled above a binary operator if: (1) a left-side relation of the GroupBy operator has a key; and (2) a predicate of the GroupBy operator does not use a result of the GroupBy operator.

15. A client-server relational database system, comprising:

a client computer connected to a server computer, wherein data from the client computer is encrypted by the client computer and hosted by the server computer, the encrypted data is operated upon by the server computer to produce an intermediate results set, the intermediate results set is sent from the server computer to the client computer where it is operated upon by the client computer and then returned to the server computer where it is further operated upon by the server computer before being sent again from the server computer to the client computer in order to produce actual results.

16. A client-server relational database system, comprising:

a server computer connected to a client computer, wherein data from the client computer is encrypted by the client computer and hosted by the server computer, the encrypted data is operated upon by the server computer to produce an intermediate results set, the intermediate results set is sent from the server computer to the client computer where it is operated upon by the client computer and then returned to the server computer where it is further operated upon by the server computer before being sent again from the server computer to the client computer in order to produce actual results.

17. A method of performing computations on encrypted data stored on a computer system, comprising:

encrypting data at client computer;

hosting the encrypted data on a server computer;

operating upon the encrypted data at the server computer to produce an intermediate results set;

transferring the intermediate results set from the server computer to the client computer;

operating upon the transferred intermediate results set at the client computer to generate an updated intermediate results set;

re-encrypting the updated intermediate results set at the client computer;

transferring the re-encrypted intermediate results set to the server computer;

operating upon the transferred intermediate results set at the server computer to generate a new intermediate results set;

transferring the new intermediate results set from the server computer to the client computer; and

producing actual results from the transferred new intermediate results set at the client computer.

18. The method of claim 17, wherein the client computer decrypts the intermediate results set, performs one or more operations on the decrypted intermediate results set to generate an updated intermediate results set, re-encrypts the updated

intermediate results set, and returns the re-encrypted intermediate results set to the server computer.

19. The method of claim 18, wherein the operations comprise logical comparison operations.

20. The method of claim 18, wherein the operations comprise filtering operations.

21. The method of claim 18, wherein the operations comprise sorting operations.

22. The method of claim 17, wherein the server computer executes a round-trip filtering operator that specifies when the intermediate results set is sent from the server computer to the client computer to be operated upon by the client computer and then returned to the server computer to be operated upon by the server computer.

23. The method of claim 22, wherein the round-trip filtering operator sends maybe tuples from the server computer to the client computer, the client computer filters out certain tuples from the maybe tuples, and the server computer receives back only certain tuples from the client computer.

24. The method of claim 22, wherein the server receives a query from the client computer, generates a plurality of query execution plans having different placements of the

round-trip filtering operator, and chooses one of query execution plans that optimizes placement of the round-trip filtering operator.

25. The method of claim 22, wherein the server executes a last-trip decryption operator that specifies when the intermediate results set is sent from the server computer to the client computer in order to produce actual results.

26. The method of claim 25, wherein the server computer receives a query from the client computer, generates a plurality of query execution plans having different placements of the last-trip decryption operator, and chooses one of query execution plans that optimizes placement of the last-trip decryption operator.

27. The method of claim 26, wherein the server computer optimizes placement for the round-trip filtering operators and then optimizes placement for the last-trip decryption operator.

28. The method of claim 25, wherein the last-trip decryption operator can be pulled up above any unary and binary operator in a query tree, except for a GroupBy operator.

29. The method of claim 28, wherein the GroupBy operator can be pulled above a unary operator other than another GroupBy operator if and only if all columns used in the unary operator are functionally computed by grouping columns of an input relation.

30. The method of claim 29, wherein the GroupBy operator can be pulled above a binary operator if: (1) a left-side relation of the GroupBy operator has a key; and (2) a predicate of the GroupBy operator does not use a result of the GroupBy operator.

31. An article of manufacture embodying logic for performing computations on encrypted data stored on a computer system, the logic comprising:

encrypting data at client computer;

hosting the encrypted data on a server computer;

operating upon the encrypted data at the server computer to produce an intermediate results set;

transferring the intermediate results set from the server computer to the client computer;

operating upon the transferred intermediate results set at the client computer to generate an updated intermediate results set;

re-encrypting the updated intermediate results set at the client computer;

transferring the re-encrypted intermediate results set to the server computer;

operating upon the transferred intermediate results set at the server computer to generate a new intermediate results set;

transferring the new intermediate results set from the server computer to the client computer; and

producing actual results from the transferred new intermediate results set at the client computer.

32. The article of claim 31, wherein the client computer decrypts the intermediate results set, performs one or more operations on the decrypted intermediate results set to generate an updated intermediate results set, re-encrypts the updated intermediate results set, and returns the re-encrypted intermediate results set to the server computer.

33. The article of claim 32, wherein the operations comprise logical comparison operations.

34. The article of claim 32, wherein the operations comprise filtering operations.

35. The article of claim 32, wherein the operations comprise sorting operations.

36. The article of claim 31, wherein the server computer executes a round-trip filtering operator that specifies when the intermediate results set is sent from the server computer to the client computer to be operated upon by the client computer and then returned to the server computer to be operated upon by the server computer.

37. The article of claim 36, wherein the round-trip filtering operator sends maybe tuples from the server computer to the client computer, the client computer filters out certain tuples from the maybe tuples, and the server computer receives back only certain tuples from the client computer.

38. The article of claim 36, wherein the server receives a query from the client computer, generates a plurality of query execution plans having different placements of the round-trip filtering operator, and chooses one of query execution plans that optimizes placement of the round-trip filtering operator.

39. The article of claim 36, wherein the server executes a last-trip decryption operator that specifies when the intermediate results set is sent from the server computer to the client computer in order to produce actual results.

40. The article of claim 39, wherein the server computer receives a query from the client computer, generates a plurality of query execution plans having different placements of the last-trip decryption operator, and chooses one of query execution plans that optimizes placement of the last-trip decryption operator.

41. The article of claim 40, wherein the server computer optimizes placement for the round-trip filtering operators and then optimizes placement for the last-trip decryption operator.

42. The article of claim 39, wherein the last-trip decryption operator can be pulled up above any unary and binary operator in a query tree, except for a GroupBy operator.

43. The article of claim 42, wherein the GroupBy operator can be pulled above a unary operator other than another GroupBy operator if and only if all columns used in the unary operator are functionally computed by grouping columns of an input relation.

44. The article of claim 43, wherein the GroupBy operator can be pulled above a binary operator if: (1) a left-side relation of the GroupBy operator has a key; and (2) a predicate of the GroupBy operator does not use a result of the GroupBy operator.